ARL-MR-0892 • June 2015

US Army Research Laboratory

Postprocessing of Voxel-Based Topologies for Additive Manufacturing Using the Computational Geometry Algorithms Library (CGAL)

by Raymond A Wildman

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **JUN 2015**	2. REPORT TYPE	3. DATES COVERED **00-10-2014 to 00-11-2014**

4. TITLE AND SUBTITLE **Postprocessing of Voxel-Based Topologies for Additive Manufacturing Using the Computational Geometry Algorithms Library (CGAL)**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **US Army Research Laboratory,,ATTN: RDRL-WMM-B,,Aberdeen Proving Ground,,MD, 21005-**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Postprocessing of 3-dimensional (3-D) topologies that are defined as a set of voxels using the Computational Geometry Algorithms Library (CGAL) is discussed. Most topology optimization algorithms return results in a 3-D voxel or 2-dimensional (2-D) pixel representation, which cannot be directly used for additive manufacturing (AM). AM typically uses a surface description of the topology in the stereolithography (STL) format, discretized using triangles. As such, any results from voxel-based topology optimization algorithms must be post-processed before 3-D printing. CGAL is a set of computational geometry algorithms, several of which are suited to the task. The work flow described in this report involves first defining a set of points in space from the voxels, either using their center points or corners. Next, a surface is generated by computing an -hull of the point set. Next, surface simplification can be used to reduce the complexity of the surface without losing any detail. Finally, surface subdivision algorithms, such as Catmull-Clark or Doo-Sabin, can be used to smooth the resulting surface.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**28**	

ARL-MR-0892 ● June 2015

US Army Research Laboratory

Postprocessing of Voxel-Based Topologies for Additive Manufacturing Using the Computational Geometry Algorithms Library (CGAL)

by Raymond A Wildman
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* June 2015	**2. REPORT TYPE** Final		**3. DATES COVERED (From - To)** October 2014-November 2014
4. TITLE AND SUBTITLE Postprocessing of Voxel-Based Topologies for Additive Manufacturing Using the Computational Geometry Algorithms Library (CGAL)			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Raymond A Wildman			**5d. PROJECT NUMBER** AH84
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-WMM-B Aberdeen Proving Ground, MD 21005-5066			**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-MR-0892
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			**10. SPONSOR/MONITOR'S ACRONYM(S)**
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			
13. SUPPLEMENTARY NOTES primary author's email: <raymond.a.wildman.civ@mail.mil>.			
14. ABSTRACT Postprocessing of 3-dimensional (3-D) topologies that are defined as a set of voxels using the Computational Geometry Algorithms Library (CGAL) is discussed. Most topology optimization algorithms return results in a 3-D voxel or 2-dimensional (2-D) pixel representation, which cannot be directly used for additive manufacturing (AM). AM typically uses a surface description of the topology in the stereolithography (STL) format, discretized using triangles. As such, any results from voxel-based topology optimization algorithms must be post-processed before 3-D printing. CGAL is a set of computational geometry algorithms, several of which are suited to the task. The work flow described in this report involves first defining a set of points in space from the voxels, either using their center points or corners. Next, a surface is generated by computing an α-hull of the point set. Next, surface simplification can be used to reduce the complexity of the surface without losing any detail. Finally, surface subdivision algorithms, such as Catmull-Clark or Doo-Sabin, can be used to smooth the resulting surface.			
15. SUBJECT TERMS geometry processing, additive manufacturing, topology optimization, CGAL			

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Raymond A Wildman
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	28	**19b. TELEPHONE NUMBER (Include area code)** 410-306-2232

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

The author wishes to sincerely thank Marc Pepi, who originally posed the example problem.

1. Introduction

Topology optimization is a maturing field that has resulted in a new way to engineer structures.[1] Essentially, given a set of loads and boundary conditions, topology optimization can deliver an optimal structure using only a user specified amount of material. One drawback of topology optimization is that optimized solutions are sometimes difficult to manufacture due to their complexity; however, additive manufacturing allows for the flexibility needed to produce optimized structures.[2] Contrary to traditional manufacturing methods, additive manufacturing proceeds by the deposition of a material, so that a structure is built up by layers.

Typically, additive manufacturing devices (3-dimensional [3-D] printers, e.g.), use the stereolithography (STL) file format, which is a surface description of an object. Unfortunately, many topology optimization methods in use today use a volumetric description of a structure, necessitating postprocessing if additive manufacturing is to be used. Worse yet, topology optimization also typically uses a continuous parameter to describe the structure in a set of pixels or voxels (3-D pixels), meaning that there is no definitive notion of a surface. The use of pixels or voxels also means that the result can be somewhat resolution-dependent. Each of these issues will be addressed in the following methodology.

In this report, a topology optimization postprocessing method is described that uses the Computational Geometry Algorithms Library (CGAL).[3] CGAL is a C++-based library that includes a large number of geometry processing algorithms. The method detailed here will focus on 3 algorithms: α-shapes, mesh simplification, and subdivision surfaces. First, α-shapes provide a method for obtaining a surface from a set of points in space. Second, mesh simplification is used to reduce the overall file size of the resulting geometry. Third, subdivision can be optionally used to smooth a surface. In addition to these 3 geometric algorithms, a set of points must be generated from the result of the topology optimization scheme. The method for doing so is straightforward, though it involves a user-specified threshold for the inclusion of points.

The remainder of the report is organized as follows: In Section 2 the overall methodology is presented along with a 2-dimensional (2-D) example. Next, Section 3 presents the postprocessing of a 3-D example. Finally, Section 4 concludes the report.

2. Methodology

In this section, a CGAL-based topology postprocessing methodology is described in detail. The postprocessing method proceeds in 5 steps: Topology optimization, point cloud generation, α-shape generation, mesh simplification, and (optionally) mesh subdivision. First, topology optimization is reviewed using the standard linear elastic method in Section 2. Next, generation of a point cloud representation is discussed. This point cloud representation is then converted to a surface using alpha-shapes as described in Section 2. Mesh simplification is next applied to reduce the size of the resulting data without altering the geometry. The resulting geometry of the the previous steps can be somewhat nonsmooth (depending on the resolution of the original optimization), so a smoothing method is discussed that uses subdivision surfaces such as Catmull-Clark or Doo-Sabin.

2.1 Topology Optimization

Topology optimization applied to structural problems using a voxel/pixel geometry representation coupled with a local optimization method is summarized in the well-known book by Bendsøe and Sigmund.[1] The basic idea is to discretize the area of interest into a set of cubes or squares and optimize for a continuous range of material properties rather than discrete values. The resolution of the optimization is given by the number of voxels in each dimension: N_{x},N_{y}, and N_{z}. The continuous range of material properties facilitates the use of a local, gradient-based optimizer, and can be specified as

$$E\left(x,y,z\right) = p\left(x,y,z\right)E_{\text{max}}, \tag{1}$$

where $E\left(x,y,z\right)$ is the elastic modulus at a point $\left(x,y,z\right)$ (assumed to be the voxel's center point), E_{max} is the maximum elastic modulus, and $0 < p \leq 1$ is the optimization parameter where $p = 1$ indicates presence of material and $p = 0$ indicates void. (In practice, $p = 0$ is not strictly allowed, as elements with zero stiffness lead to a singular stiffness matrix. Instead, some small value p_{min} is used, though $p = 0$ will be stated throughout for simplicity.)

After discretization, boundary conditions are specified in terms of forces and displacements. The boundary conditions, along with the geometry of the region of interest and desired volume fraction, determine the solution. In other words, a structure optimized for compression will be much different than one designed for shear loading.

Ultimately, the solution to a given problem is specified as a set of voxels or pixels, each with a real-valued number p specifying the elastic modulus at that voxel. Penalization methods are applied to restrict, as much as possible, the solution to discrete values of elastic moduli (i.e., $p = 0$ or $p = 1$), though near a structure's edges, nondiscrete values are typical. The lack of discrete solutions for p introduces the first approximation necessary for postprocessing of optimized solutions: A threshold value for p must be chosen that specifies which voxels to include in the solution.

As an example of the above process in 2-D, consider a square region in which we wish to optimize a structure for pure compression, applied along the top edge. The displacements are fixed on the bottom surface and the remaining sides are assumed to be traction free. The basic topology optimization process outlined above leads to the solution given in Fig. 1 with a resolution of $N_{\text{x}} = 50$ and $N_{\text{y}} = 50$. In Fig. 1, black indicates $p = 1$ and white indicates $p = 0$. The blurriness seen in the transition from black to white are intermediate values of p, which must be dealt with using a threshold procedure, described in the next subsection.

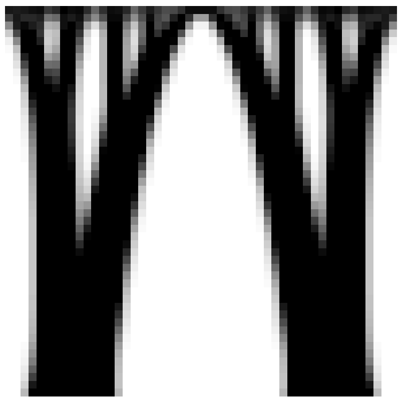

Fig. 1 An example of a 2-D topology optimization result, optimized for compression

2.2 Point Cloud Generation

After an optimal design is computed, several postprocessing steps are necessary to generate a 3-D-printable data file. First, a basic threshold procedure is used to eliminate nondiscrete values of p. Given a solution **P**, where **P** is a 3-D array containing the optimal values of p at a set of discrete voxels:

$$\mathbf{P}_{ijk} = p\left(x_i, y_j, z_k\right). \tag{2}$$

A discrete approximation $\mathbf{P}^*$ of the solution can then be formed from **P** by

$$\mathbf{P}^*_{ijk} = \begin{cases} 0, & \mathbf{P}_{ijk} < \tau \\ 1, & \mathbf{P}_{ijk} \geq \tau \end{cases}, \tag{3}$$

where τ is some threshold value, which will later be set to $1/2$. Again, note that $\mathbf{P}^*$ is not an optimal solution according to the original problem statement, though it should well approximate the optimal solution. Now, the voxel representation can be converted to a set of points in one of 2 ways: Using either the center of the voxel or the corners. In this report, the corners will be used. To proceed, first the voxel side length is assumed to be the same for each dimension. In other words, the width, length, and height of the voxel are equal and given by Δ. Now, given the side length and array indices, a set of 8 points can be defined as

$$\begin{aligned}
\mathbf{r}^1_{ijk} &= \left(\left(i-1\right)\Delta, \left(j-1\right)\Delta, \left(k-1\right)\Delta\right), \\
\mathbf{r}^2_{ijk} &= \left(\left(i-1\right)\Delta, \left(j-1\right)\Delta, k\Delta\right), \\
\mathbf{r}^3_{ijk} &= \left(\left(i-1\right)\Delta, j\Delta, \left(k-1\right)\Delta\right), \\
\mathbf{r}^4_{ijk} &= \left(\left(i-1\right)\Delta, j\Delta, k\Delta\right), \\
\mathbf{r}^5_{ijk} &= \left(i\Delta, \left(j-1\right)\Delta, \left(k-1\right)\Delta\right), \\
\mathbf{r}^6_{ijk} &= \left(i\Delta, \left(j-1\right)\Delta, k\Delta\right), \\
\mathbf{r}^7_{ijk} &= \left(i\Delta, j\Delta, \left(k-1\right)\Delta\right), \\
\mathbf{r}^8_{ijk} &= \left(i\Delta, j\Delta, k\Delta\right),
\end{aligned} \tag{4}$$

where indices i, j, and k range from 1 to their respective limits, N_x, N_y, and N_z. Finally, the complete set of points is given by the corner points of the entries of $\mathbf{P}^*$ that have a value of 1. For simplicity, this set of points can be described as a $8N_\text{x}N_\text{y}N_\text{z}$-by-3 array **R** with rows

$$\mathbf{R}_m = \mathbf{P}^*_{ijk}\mathbf{r}^l_{ijk}, \tag{5}$$

where row $m = i + N_\text{x}\left(j-1\right) + N_\text{x}N_\text{y}\left(k-1\right) + N_\text{x}N_\text{y}N_\text{z}\left(l-1\right)$ and index l ranges from 1 to 8, as in Eq. 4. (Note that Einstein summation is *not* intended throughout.)

A strict interpretation of Eq. 5 leads to duplicate points for 2 reasons: First, any element of $\mathbf{P}^*$ that is zero (void) yields a point at the origin $(0,0,0)$. Second, any adjacent material points will share some corner nodes, e.g., $\mathbf{r}^1_{ijk} = \mathbf{r}^8_{i-1,j-1,k-1}$. It is important then to remove any duplicate points from $\mathbf{R}$ before proceeding.

As a 2-D example, the result given in Fig. 1 was converted to a point set using the above procedure and a threshold value of $\tau = 0.5$ and a side length of $\Delta = 1$. The result is given in Fig. 2.

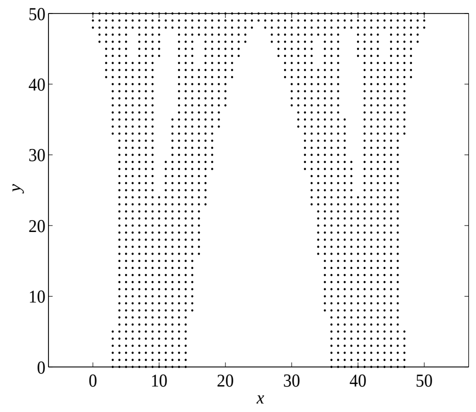

Fig. 2 Point set representation of the result shown in Fig. 1

2.3 Alpha-Shape Point Processing

Now that a point representation of the optimized topology has been defined, it can be processed into a surface. There are several methods available to accomplish this goal, though here we use the concept of α-shapes.[4,5] Generally, an α-shape is a linear approximation of the surface of a dense set of points. More specifically, an α-shape is a generalization of the convex hull of a set of points and is related to the Delaunay triangulation (or tesselation in 3-D) of the point set. In 2-D, the α-shapes of a set of points can be thought of as the surface resulting from attempting to slide a disk of radius $\sqrt{\alpha}$ between each pair of points. Pairs of points through which the disk cannot pass become edges of the surface. Finally, only edges defining topological features such as the outer surfaces and any internal holes are retained.

Given that we now have a dense set of points, an α-shape is a convenient way to generate a linear surface. The point set $\mathbf{R}$ can then be processed using the α-shape algorithms in CGAL; however, the choice of α changes the resulting surface.

Consider, a very large α (that approaches infinity) is equivalent to the convex hull of the set of points. On the other hand, a very small α (that approaches zero) will pass through all pairs of points, not generating a surface at all!

It is then apparent that the parameter α (or the squared radius of the largest line segment included in a 2-D model) highly influences the resulting surface. Fortunately, we can bound α in a reasonable way given the discretization of the original optimization problem. A lower bound is then

$$\alpha > \frac{\Delta^2}{4}, \tag{6}$$

given that if α is any smaller, the result will not include any defining surface as discussed above. There is some flexibility in the choice of upper bound of α: Choosing α to be too large will result in interior holes being removed, while choosing too small of an α leads to a rough surface. Further, while α is a continuous parameter, the α-shapes associated with a given set of points is actually a discrete, finite set, so that each α-shape can be examined if desired.

As an example, the point set shown in Fig. 2 is shown processed with several values of α, in increasing order from Figs. 3 to 6. While the lowest possible α in this case leads to a somewhat rough shape, higher values of α can lead to loss of detail, especially in small holes. Fortunately, subdivision can be applied to help with surface roughness.

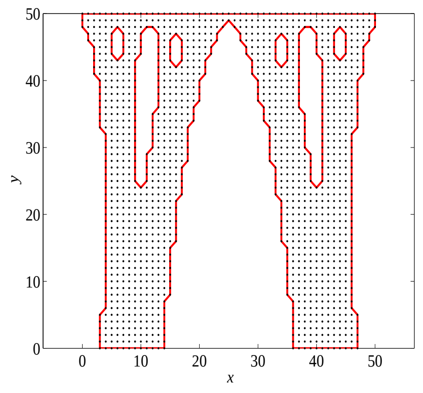

Fig. 3 **α-shape representation of the result shown in Fig. 2**

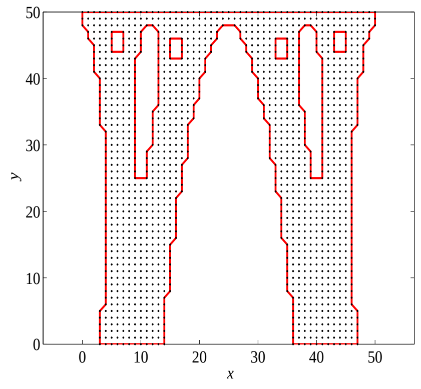

Fig. 4 **α-shape representation of the result shown in Fig. 2**

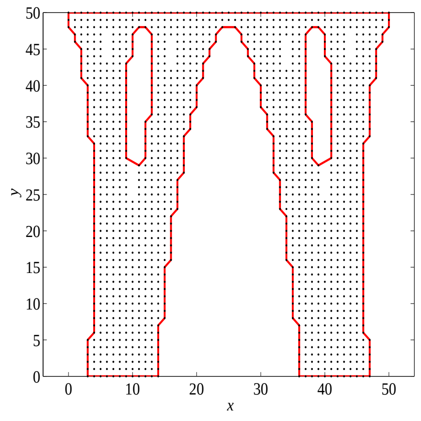

Fig. 5 **α-shape representation of the result shown in Fig. 2**

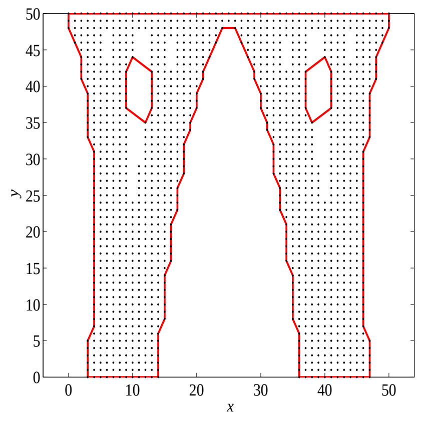

Fig. 6 **α-shape representation of the result shown in Fig. 2**

2.4 Mesh Simplification

Mesh simplification can next be applied to reduce the complexity of the surface representation without losing any detail. The essential function of mesh simplification is to remove and join any coplanar facets (or collinear line segments in 2-D). This is also an important step if subdivision is applied, as subdivision works best on meshes without small, coplanar facets. While the removal of only coplanar facets results in an identical geometry description, the mesh simplification algorithms in CGAL will reduce the complexity of the mesh to a user specified level, ultimately coarsening the mesh. In this process, the idea is to retain rough features while reducing complexity. For the application here, we simply want to reduce the number of facets without changing the overall geometry. One method for this is to simply compute the surface area of or the volume enclosed by the original and simplified meshes, ensuring that they are identical. A good starting point is to attempt to reduce the number of edges by 50%. The CGAL user manual entry for surface mesh simplification can be found in Cacciola 2015.[6]

To demonstrate the process in 2-D, we can remove all collinear line segments from the example above. While this is an oversimplification of the process in 3-D, it is the goal of our use of the algorithm (i.e., removal of coplanar facets). Fig. 7 shows the result of the α-shape algorithm described above (see Fig. 3), with the nodes shown in red and the line segments in blue. Clearly, this surface contains many collinear line segments, which are redundant for the purposes of representing the surface. The collinear segments can be removed by simply iterating around the spline, and joining any adjacent segments with the same slope or angle. Fig. 8 gives the simplified form.

2.5 Subdivision

As a final (optional) step, surface subdivision can be applied to reduce surface roughness. Surface subdivision is a smoothing process that can be applied to linear surface representations.[7] The process is similar in 2-D and 3-D and essentially involves inserting new vertices around corners iteratively, until a surface is smoothed to the desired level. In that sense, a recursively applied subdivision method will converge to an approximately smooth surface. In 3-D, several options are available in CGAL including Doo-Sabin,[8,9] Catmull-Clark,[10] loop subdivision, and the so-called $\sqrt{3}$ subdivision. The differences in the methods in 3-D lies in how the polygonal facets of the mesh are split.

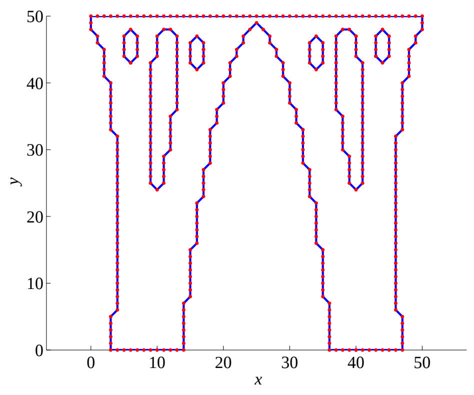

Fig. 7 Linear spline representation before simplification

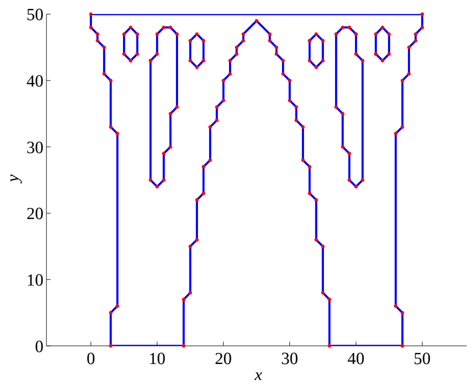

Fig. 8 Linear spline representation after simplification

In 2-D, the process is simple, but illustrative. Consider a closed linear spline, defined by a set of N vertices arranged in an N-by-2 array $\mathbf{V}$. Subdivision proceeds by replacing each vertex $\mathbf{V}_i$ with 2 new points located along the 2 adjacent line segments. The location on each line segment is defined as a given, constant percentage of the total length of the line segment, denoted t. New vertices $\mathbf{V}_i^1$ and $\mathbf{V}_i^2$ associated with original vertex $\mathbf{V}_i$ are then given by

$$\begin{aligned}\mathbf{V}_i^1 &= \mathbf{V}_i + t\left(\mathbf{V}_{i-1} - \mathbf{V}_i\right), \\ \mathbf{V}_i^2 &= \mathbf{V}_i + t\left(\mathbf{V}_{i+1} - \mathbf{V}_i\right).\end{aligned} \tag{7}$$

Recall that each vertex is replaced with two new vertices, so that each step of subdivision results in a doubling of the number of vertices. In other words, if the r denotes the current iteration of subdivision, the number of vertices will be $2^r N$. Finally, we may not want to smooth all vertices, such as those associated with boundary or loading conditions. For such vertices we can simply not apply the subdivision process.

As an example, 2-D subdivision was applied several times to the surface shown in Fig. 3 with $t = 0.25$. Figures 9 and 10 show the result of 1 and 4 iterations of subdivision applied to the surface shown in Fig. 3, respectively. Subdivision is not a perfect solution: As can be seen in the figures, roughness is still apparent at the resolution of the original, pixel-based solution.

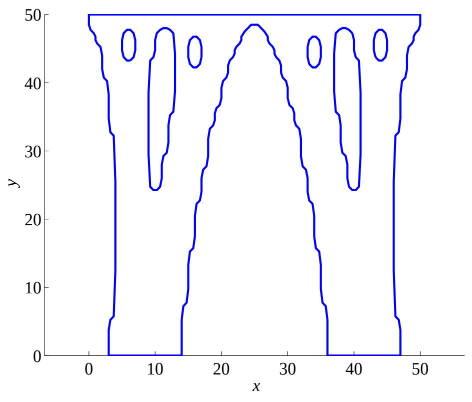

Fig. 9 One iteration of subdivision applied to the surface in Fig. 3

Finally, the processed surface is compared with the original, pixel-based solution from the topology optimization in Fig. 11. It is clear from the comparison that some features may be missing, and some areas where sharp corners or cusps may have been intended are rounded out.

2.6 Summary

The aim of the method presented here is to begin with a standard, voxel-based topology optimization scheme and end with an STL file, ready for use in a 3-D printer or other additive manufacturing device. Given that aim, a final summary is discussed here, with additional technical details that may be helpful in reproducing the method.

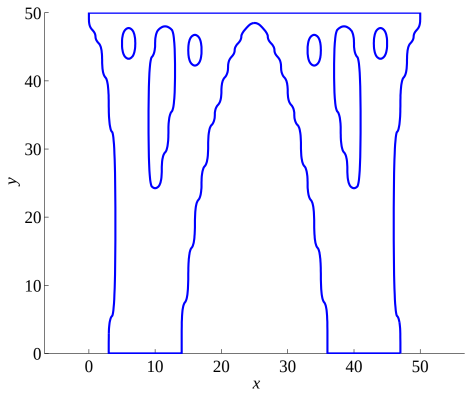

Fig. 10 Four iterations of subdivision applied to the surface in Fig. 3

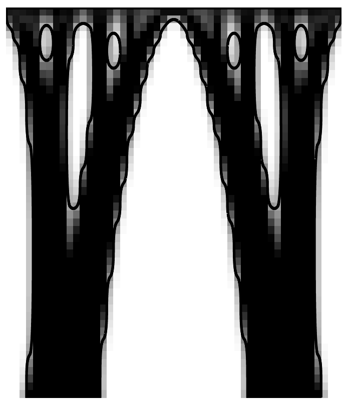

Fig. 11 Comparison of pixel solution and processed surface

Many basic topology optimization codes can be found on the Internet, the most well-known being the "99 line topology optimization code written in MATLAB".[11] Three-dimensional versions are also available for MATLAB.[12] After a structure has been optimized, the first step is again to convert the voxel representation to a discrete set of points in space. The method described in Section 2 can be used and only requires a user-specified tolerance. The output of this method is simply a list of 3-D coordinates that can be saved as a text file.

Next, CGAL is used to generate a surface using α-shape processing. The CGAL download package contains example programs, one of which demonstrates the generation of α-shapes. This example program can be simply modified to read the text file generated in the previous step. The output from CGAL is a surface in the object file format (OFF), a basic text-based description of the surface that includes vertices and connectivity information for the polygonal facets. This OFF file can be read directly into the surface mesh simplification algorithm. Again, an example program is available in the CGAL download and only slight modifications are necessary to accommodate an arbitrary input file. This function also requires a stopping criterion in terms of the percent of edges removed. A good starting point is 50%. Finally, CGAL's subdivision algorithms also use an OFF file for input and output.

The final step is to convert the OFF file into an STL file. This can be accomplished with MATLAB using the following procedure: First, read in the OFF file and save the data as 2 matrices. One matrix contains the vertex coordinates in an m-by-3 matrix and the second contains the connectivity information of the surface as a n-by-3 matrix. These matrices can then be written to the STL format using a downloadable MATLAB function.[13]

The table lists the algorithm, implementation language, and input and output file formats for each step of the postprocessing method.

Table Summary of CGAL-based postprocessing procedure

Step	Algorithm	Language	User input	Input Format	Output Format
1	Optimization	MATLAB	N_x, N_y, BCs	NA	Matrix
2	Point conversion	MATLAB	Threshold	Matrix	Text file
3	α-shape	CGAL (C++)	α	Text file	OFF
4	Simplification	CGAL (C++)	$\%$ reduction	OFF	OFF
5	Subdivision	CGAL (C++)	No. of iterations	OFF	OFF
6	STL Conversion	MATLAB	NA	OFF	STL

3. 3-D Example

To demonstrate the method presented above, a 3-D structure was optimized for compression. The structure is to fit into a cube and will ultimately be used as a unit cell in a repeated structure. Though the intention is for the structure to be optimized for compression, additional in-plane loads were also applied to make the structure more robust to buckling. (Because a linear elastic forward solver is used, the optimization cannot explicitly account for buckling and failure.) Accordingly, the structure was optimized for 5 loads:

$$\begin{aligned}
\mathbf{f}_1(x, y, z = h) &= -\hat{\mathbf{z}}, \\
\mathbf{f}_2(x, y, z = h) &= \frac{1}{2}\hat{\mathbf{x}}, \\
\mathbf{f}_3(x, y, z = h) &= -\frac{1}{2}\hat{\mathbf{x}}, \\
\mathbf{f}_4(x, y, z = h) &= \frac{1}{2}\hat{\mathbf{y}}, \\
\mathbf{f}_5(x, y, z = h) &= -\frac{1}{2}\hat{\mathbf{y}},
\end{aligned} \tag{8}$$

all applied to the top surface, or $z = h$ plane, where h is the height. Zero displacement boundary conditions on each component of the displacement were used on the bottom surface, or $z = 0$ plane. The volume fraction was set to 0.25 and a resolution of $N_\text{x} = N_\text{y} = N_\text{z} = 20$ was used. The optimization problem was solved using the MATLAB code listed in Liu and Tovar 2015.[12]

Step 2 (point conversion, see the above table) was completed with a threshold value of 0.5. After the points were generated, they were saved to a text file in 3 columns, one for each component. This file was then used in the α-shape algorithm from CGAL. CGAL also provides a function to return the optimal α-shape for a given number of connected components, which for our application is one single component. This option was used to generate the structure shown in Fig. 12.

For the application considered here, the result shown in Fig. 12 was used as a unit cell, and a larger structure was generated by repeating the unit cell along the x, y, and z axes 5 times on each axis. This repetition was completed using the point cloud representation, and a surface was generated again using the α-shape algorithm.

Next, the resulting surface was simplified using CGAL's mesh simplification algorithm. In this case, a minimum reduction percentage of 47.5%, reducing the OFF file size from 26 MB to 8 MB without losing any geometric detail. The resulting structure is shown in Fig. 13.

Finally, one iteration of subdivision was applied using CGAL's loop subdivision algorithm. The algorithm was altered so that facets, edges, and vertices lying on either the $z = 0$ or $z = 0.04$ planes were not subdivided. A detail view of the bottom corner of the resulting structure is shown in Fig. 14 and for comparison a similar view of the original structure is shown in Fig. 15.

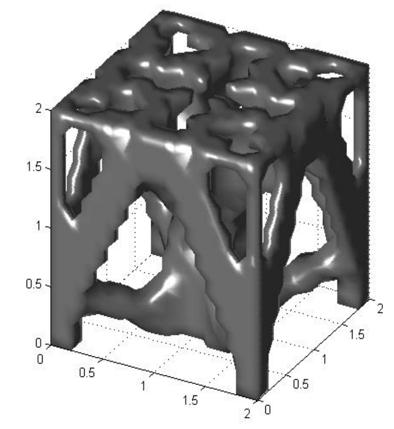

Fig. 12 Unit cell after α-shape processing

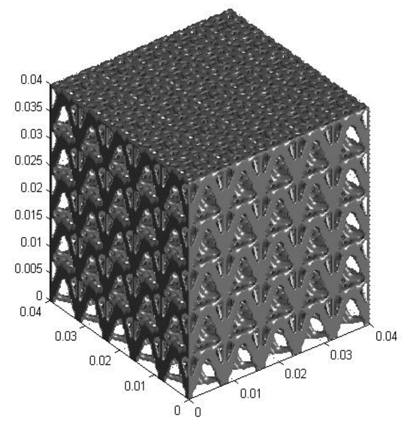

Fig. 13 Final compression structure

4. Conclusions

A postprocessing methodology has been presented for results of topology optimization methods that give solutions in terms of continuous values on a fixed grid (voxel-based representation). Additive manufacturing of topologically optimized structures is a promising area of research, though direct printing of voxel solutions is not possible. Options for postprocessing include the method presented here, based on computational geometry algorithms implemented in CGAL, and also isosurfaces. An isosurface approach attempts to fit a surface description to the discretized data at a given threshold. One drawback of this approach is that no consideration to component connectivity is given by isosurface algorithms. As discussed

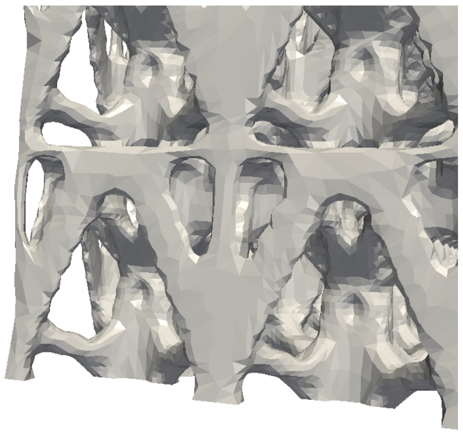

Fig. 14 Detail view of the final compression structure with subdivision

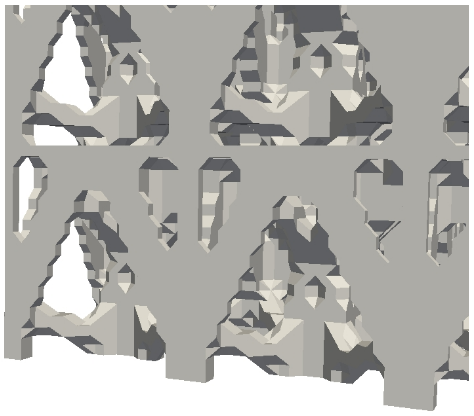

Fig. 15 Detail view of the final compression structure without subdivision

above, an approach based on α-shapes can ensure that only one connected component is generated. One advantage of an isosurface approach is that its surfaces tend to be smoother than those generated by an α-shape algorithm, reducing the need for simplification and subdivision.

Regardless of the approach used, postprocessing of topology optimization results is an important area of study, and more work is necessary to ensure that the processed surfaces retain the optimal properties of the voxel representation. Recall, the voxel representation allows for continuous material properties while any post-processed surface is made up of discrete materials. This will likely change both the volume fraction specified in the optimization problem and, more importantly, the mechanical properties.

5. References

1. Bendsøe M, Sigmund O. Topology optimization: Theory, methods and applications. Springer; 2003. (Engineering Online Library).

2. Brackett D, Ashcroft I, Hague R. 22nd annual international solid freeform fabrication symposium. 2011.

3. The computational geometry algorithms library. 2015 [accessed 2015 May 18]. http://www.cgal.org/.

4. Edelsbrunner H, Mücke EP. Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG). 1994;13(1):43–72.

5. Da TKF, Loriot S, Yvinec M. Cgal 4.6 - 3d alpha shapes. 2015 [accessed 2015 May 18]. http://doc.cgal.org/latest/Alpha_shapes_3/index.html#Chapter_3D_Alpha_Shapes.

6. Cacciola F. Cgal 4.6 - triangulated surface mesh simplification. 2015 [accessed 2015 May 18]. http://doc.cgal.org/latest/Surface_mesh_simplification/index.html#Chapter_Triangulated_Surface_Mesh_Simplification.

7. Shiue LJA. Cgal 4.6 - 3d surface subdivision methods. 2015 [accessed 2015 May 18]. http://doc.cgal.org/latest/Subdivision_method_3/index.html#Chapter_3D_Surface_Subdivision_Methods.

8. Doo D, Sabin M. Behaviour of recursive division surfaces near extraordinary points. Computer-Aided Design. 1978;10(6):356–360.

9. Doo D. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In: Proceedings on Interactive Techniques in Computer Aided Design; Vol. 157; p. 165.

10. Catmull E, Clark J. Recursively generated b-spline surfaces on arbitrary topological meshes. Computer-aided design. 1978;10(6):350–355.

11. Sigmund O. A 99 line topology optimization code written in matlab. 2015 [accessed 2015 May 18]. http://www.topopt.dtu.dk/?q=node/2.

12. Liu K, Tovar A. Top3d - an efficient 3d topology optimization program. 2015 [accessed 2015 May 18]. https://top3dapp.com/.

13. Holcombe S. stlwrite. 2015 [accessed 2015 May 18]. http://www.mathworks.com/matlabcentral/fileexchange/20922-stlwrite-filename--varargin-.

1	DEFENSE TECHNICAL
(PDF)	INFORMATION CTR
	DTIC OCA
2	DIRECTOR
(PDF)	US ARMY RESEARCH LAB
	RDRL CIO LL
	IMAL HRA MAIL & RECORDS MGMT
1	GOVT PRINTG OFC
(PDF)	A MALHOTRA
	<u>ABERDEEN PROVING GROUND</u>
22	DIR USARL
(PDF)	RDRL WM
	B FORCH
	J MCCAULEY
	RDRL WML H
	B SCHUSTER
	RDRL WMM
	J BEATTY
	R DOWDING
	RDRL WMM A
	L HOLMES
	R RODRIGUEZ
	RDRL WMM B
	G GAZONAS
	D HOPKINS
	B LOVE
	B POWERS
	R WILDMAN
	RDRL WMM D
	R CARTER
	B MCWILLIAMS
	M PEPI
	L VARGAS
	A GAYNOR
	RDRL WMM F
	T SANO
	M TSCHOPP
	RDRL WMM G
	J ANDZELM
	RDRL WMP B
	M SCHEIDLER
	A SOKOLOW

INTENTIONALLY LEFT BLANK.